FIŞA DISCIPLINEI

Programare orientată obiect

Anul universitar 2025-2026

1. Date despre program

1.1. Instituţia de învăţământ superior	Universitatea Babeş-Bolyai Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică şi Informatică
1.3. Departamentul	Departamentul de informatică
1.4. Domeniul de studii	Informatică
1.5. Ciclul de studii	Licenţă
1.6. Programul de studii / Calificarea	Informatică engleză
1.7. Forma de învăţământ	Cu frecventa

2. Date despre disciplină

2.1. Denumirea disciplinei	**Programare orientată obiect / Object oriented programming**					Codul disciplinei	**MLE5006**
2.2. Titularul activităţilor de curs	Conf. dr. Bocicor Maria Iuliana						
2.3. Titularul activităţilor de seminar	Conf. dr. Bocicor Maria Iuliana						
2.4. Anul de studiu	1	2.5. Semestrul	2	2.6. Tipul de evaluare	E	2.7. Regimul disciplinei	Obligatorie

3. Timpul total estimat (ore pe semestru al activităţilor didactice)

3.1. Număr de ore pe săptămână	**5**	din care: 3.2. curs	**2**	3.3. seminar/ laborator/proiect	**1 sem+ 2 lab**
3.4. Total ore din planul de învăţământ	70	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	**14 + 28**
Distribuţia fondului de timp pentru studiul individual (SI) şi activităţi de autoinstruire (AI)					**ore**
Studiul după manual, suport de curs, bibliografie şi notiţe (AI)					20
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					5
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii şi eseuri					19
Tutoriat (consiliere profesională)					4
Examinări					7
Alte activităţi					
3.7. Total ore studiu individual (SI) şi activităţi de autoinstruire (AI)					
3.8. Total ore pe semestru					
3.9. Numărul de credite					

4. Precondiţii (acolo unde este cazul)

4.1. de curriculum	Fundamentele Programării
4.2. de competenţe	Cunoştinţe medii de programare într-un limbaj de programare de nivel înalt

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Sală de curs cu videoproiector
5.2. de desfăşurare a seminarului/ laboratorului	Sală de laborator cu calculatoare dotate cu limbajul de programare C++ şi Qt

6.1. Competenţele specifice acumulate[1]

[1] Se poate opta pentru competenţe sau pentru rezultatele învăţării, respectiv pentru ambele. În cazul în care se alege o singură variantă, se va şterge tabelul aferent celeilalte opţiuni, iar opţiunea păstrată va fi numerotată cu 6.

Competenţe profesionale/esenţiale	• Programarea în limbaje de nivel înalt (C/C++) • Dezvoltarea şi întreţinerea aplicaţiilor informatice
Competenţe transversale	• Aplicarea regulilor de muncă organizată şi eficientă, a unor atitudini responsabile faţă de domeniul didactic-ştiinţific, pentru valorificarea creativă a propriului potenţial, cu respectarea principiilor şi a normelor de etică profesională • Utilizarea unor metode şi tehnici eficiente de învăţare, informare, cercetare şi dezvoltare a capacităţilor de valorificare a cunoştinţelor, de adaptare la cerinţele unei societăţi dinamice şi de comunicare în limba română şi într-o limbă de circulaţie internaţională

6.2. Rezultatele învăţării

Cunoştinţe	• Studentul are cunostinte necesare pentru dezvoltarea programelor şi aplicaţiilor software folosind programarea orientata obiect si limbajul de programare C++. • Studentul are abilitatea de a dezvolta, proiecta şi crea noi aplicaţii folosind bunele practici din domeniu.
Aptitudini	• Studentul este capabil să inteleaga si sa utilizeze concepte ale programarii orientate obiect pentru a dezvolta aplicatii software de complexitate medie. • Studentul este capabil să distingă între paradigmele de programare imperativă tradiţională şi cea orientată pe obiect, să explice rolul claselor ca blocuri modulare fundamentale şi să demonstreze înţelegerea conceptelor de moştenire, polimorfism, legare dinamică şi structuri generice în crearea de cod reutilizabil şi uşor de întreţinut. • Absolventul are abilitatea de a aplica reguli generale unor probleme specifice şi de a produce soluţii relevante.
Responsabilităţi şi autonomie	• Studentul are capacitatea de a lucra independent pentru a scrie programe C++ de complexitate mica/medie folosind interfete grafice si de a utiliza clase deja existente pentru constuirea sistemelor. • Studentul are deprinderile necesare pentru a dezvolta aplicaţii cu interfeţe grafice folosind şabloane arhitecturale potrivite pentru specificul aplicaţiilor cu interacţiune utilizator.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	• Să deprindă studentul cu proiectarea orientată obiect a problemelor de scară mică/mijlocie şi învăţarea limbajului de programare C++ şi crearea de interfeţe grafice utilizator în Qt.
7.2 Obiectivele specifice	• Demonstrarea diferenţei dintre proiectarea imperativă tradiţională şi proiectarea orientată obiect. • Explicarea structurilor de tip clasă ca fiind componente fundamentale, modulare. • Înţelegerea rolului moştenirii, polimorfismului, legării dinamice şi a structurilor generice în realizarea codului reutilizabil. • Explicarea şi utilizarea dezvoltării bazate pe funcţionalităţi, dezvoltarea bazată pe testare, utilizarea aserţiunilor formale şi tratarea excepţiilor. • Scrierea de programme de scară mică/mijlocie folosind C++ şi Qt. • Utilizarea claselor scrise de alţi programatori în dezvoltarea sistemelor.

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii
1. **Elemente de bază ale limbajului C.** • Elemente de bază ale limbajului C. • Elemente lexicale. Operatori. Conversii. • Tipuri de date. Variabile. Constante. • Domeniul de vizibilitate şi durata de viaţă a variabilelor. Declararea şi definirea funcţiilor. Supraîncărcarea funcţiilor. Funcţii inline.	• Expunerea interactivă • Explicaţia • Conversaţia • Exemple • Demonstraţia didactică	
2. Programare modulară în C/C++. • Funcţii. Parametri. • Pointeri si gestiunea memoriei. • Pointeri la functii • Fişiere header. Biblioteci. Implementarea modulară a TAD-urilor.		
3. Programare orientată obiect în C++. • Clase şi obiecte. • Definirea claselor. • Crearea si distrugerea obiectelor. • Supraincarcarea operatorilor. Elemente statice si prietene.		
4. Elemente de programare generica • Functii/clase parametrizate. Mecanismul de template din C++. • Containere şi iteratori – biblioteca STL. • Algoritmi STL.		
5. Moştenire • Moştenire simplă. Clase derivate. • Functii speciale in clase relativ la mostenire. • Principiul substituţiei. • Supraîncărcarea metodelor. • Moştenire multiplă. • Relaţii de specializare/generalizare – reprezentări UML.		
6. Polimorfism • Mostenire, polimorfism. • Legare statica si dinamica. • Metode virtuale. • Casting. • Clase abstracte.		
7. Stream-uri si gestiunea exceptiilor • Stream-uri input/output. • Operatori de insertie, extractie. • Formatare, manipulators, flags. • Fisiere text. • Gestiunea exceptiilor. Cod exception-safe.		
8. Gestiunea resurselor şi RAII		

• Resource Acquisition Is Initialization (RAII) • Smart pointers • RAII în STL. Smart pointers în STL.		
9. Interfețe grafice utilizator (GUI) • QT Toolkit: instalare, instrumente și module Qt • Componente grafice utilizator • Layout management • Qt Designer		
10. Elemente de programare bazată pe evenimente • Callbacks. • Evenimente: Semnale si sloturi Qt • Proiectare GUI		
11. Elemente de programare bazată pe evenimente • Șabolul MVC. • Componente grafice cu modele • Utilizarea modelelor predefinite. Implementarea modelelor personalizate. • Studiu de caz: aplicație pentru gestiunea genelor.		
12. Șabloane de proiectare • Șabloane de proiectare creaționale, structurale, comportamentale. • Exemple		
13. Șabloane de proiectare • Șablonul de proiectare Adapter. • Șablonul de proiectare Observer. • Șablonul de proiectare Iterator. • Șablonul de proiectare Composite. • Șablonul de proiectare Strategy. • Studii de caz și exemple.		
14. Recapitulare • Recapitularea celor mai importante teme discutate la curs. • Ghid de examinare.		

Bibliografie

1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995.
3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002.
4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005.
5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013.
7. C++ reference (http://en.cppreference.com/w/).
8. Qt Documentation (http://doc.qt.io/qt-6/).
9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995.

8.2 Seminar	Metode de predare	Observații
1. Probleme simple în C. Funcții. Structuri și vectori.	• Expunerea interactivă • Explicația • Conversația • Exemple • Demonstrația didactică	Seminarul este structurat sub forma a 2 ore din 2 în 2 săptămâni.
2. Programare modulară.		
3. Clase. Supraîncărcarea operatorilor. Clase cu obiecte ca date membri. Templates (vector		

dinamic).		
4. Moştenire, polimorfism		
5. Fişiere, excepţii, containere STL, iteratori, algoritmi.		
6. Interfete grafice utilizator		
7. Probleme complexe implementate pe baza diagramelor UML. Şabloane de proiectare.		

Bibliografie

1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995.
3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002.
4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005.
5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013.
7. C++ reference (http://en.cppreference.com/w/).
8. Qt Documentation (http://doc.qt.io/qt-6/).
9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995.

8.3 Laborator	Metode de predare	Observaţii
1. Instalare compilator C++ (MSVC/MinGW) şi IDE (Visual Studio/Eclipse CDT). Aspecte generale C/C++.	• Explicaţia • Conversaţia	
2. Programare simple în C.		
3. Dezvoltare software bazată pe funcţionalităţi. Arhitectură stratificată. Dezvoltare bazată pe teste. Programare modulară. (I)		
4. Dezvoltare software bazată pe funcţionalităţi. Arhitectură stratificată. Dezvoltare bazată pe teste. Programare modulară. (II)		
5. Programare orientată obiect în C++. (I)		
6. Programare orientată obiect în C++. (II)		
7. Test de laborator.		
8. Moştenire şi polimorfism.		
9. Fişiere text, excepţii. Containere STL, iteratori şi algoritmi STL.		
10. Test de laborator.		
11. GUI folosind Qt. (I)		
12. GUI folosind Qt. (II)		
13. Test de laborator.		
14. Predare laboratoare.		

Bibliografie

1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995.
3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002.
4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005.
5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013.
7. C++ reference (http://en.cppreference.com/w/).
8. Qt Documentation (http://doc.qt.io/qt-6/).
9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995.

9. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatori reprezentativi din domeniul aferent programului

- Conținutul disciplinei este în concordanță cu recomandările ACM pentru domeniul informatică.
- Cursul există în programul de studiu al universităților importante din țară și străinătate.
- Conținutul disciplinei este considerat de companiile software ca important pentru a asigura cunoștințe medii de programare orientată obiect.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	Corectitudinea și completitudinea cunoștințelor acumulate și capacitatea de a proiecta și implementa programe C++.	Examen scris (în sesiune)	30%
10.5 Seminar/laborator	Capacitatea de a proiecta, testa și depana programe C++ folosind Qt.	Evaluare practică (în sesiune)	30%
	Corectitudinea temelor de laborator si teste de laborator.	Programe, documentații. Observarea continuă pe parcursul semestrului. Teste de laborator.	40%
10.6 Standard minim de performanță			

- Fiecare student trebuie să demonstreze că a atins un nivel acceptabil de cunoaștere și înțelegere a domeniului, că este capabil să exprime cunoștințele într-o formă coerentă, că are capacitatea de a stabili anumite conexiuni și de a utiliza cunoștințele în rezolvarea unor probleme în limbajul de programare C++.
- Pentru promovare, este obligatorie prezența la minim 5 seminarii și 12 laboratoare .
- Pentru promovare este ca necesar notele pentru toate evaluarile (activitate de laborator, evaluare practica, examen scris) sa fie minim 5 și nota finală să fie minim 5.

11. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)[2]

Nu se aplică.

Data completării: 15.04.2025

Semnătura titularului de curs

Conf. Dr. Bocicor Maria Iuliana

Semnătura titularului de seminar

Conf. Dr. Bocicor Maria Iuliana

Data avizării în departament:

...

Semnătura directorului de departament

Conf.dr. Adrian STERCA

[2] Păstrați doar etichetele care, în conformitate cu *Procedura de aplicare a etichetelor ODD în procesul academic*, se potrivesc disciplinei și ștergeți-le pe celelalte, inclusiv eticheta generală pentru *Dezvoltare durabilă* - dacă nu se aplică. Dacă nicio etichetă nu descrie disciplina, ștergeți-le pe toate și scrieți "*Nu se aplică.*".